Remarks

Applicants respectfully request reconsideration of the present application in view of the foregoing amendments and the following remarks. Claims 1, 3-13, 15, 17-21, 23, 24, 26, 28, 31-34, 36, 38, and 39 are pending in the application. No claims have been allowed. Claims 1, 20, 31, and 36 are independent. Claims 1, 20, 31, and 36 have been amended.

Cited Art

The Action cites Gordon, U.S. Patent No. 6,560,774 (hereinafter "Gordon"); Inside Microsoft .NET IL Assembler (hereinafter "Microsoft-IL"); and Syme, U.S. Patent No. 7,346,901 (hereinafter "Syme").

Claim Rejections under 35 U.S.C. § 101

The Action rejects claims 1, 3-13, 15, 17-19, and 31-34 under 35 U.S.C § 101 as being directed to non-statutory subject matter. Applicants respectfully disagree. However, to expedite prosecution, Applicants have amended claims 1 and 31

Claim 1 has been amended to recite, "A method, *implemented at least in part by a computing device comprising a processing unit and memory*, …, the method comprising: *with the computing device*: instantiating … storing … ." Therefore, claim 1 (and its dependent claims) is tied to the machine (the computing device) that accomplishes the operations.

Claim 31 has been amended similarly to claim 1.

Because claims 1, 3-13, 15, 17-19, and 31-34 are directed to statutory subject matter, Applicants request that the § 101 rejection be withdrawn

Response to Examiner's Arguments

Regarding a size of a type, claim 1 recites, "wherein the classifications of types comprises a primitive type associated with a primitive type size, and wherein the primitive type size is settable to represent a constant size, the primitive type size is settable to represent a symbolic size, and the primitive type size is settable to represent an unknown size."

Regarding the type itself, claim 1 recites, "wherein one of the sub-classes representing a primitive type represents an *unknown type*, *wherein the unknown type can represent any type*,

and *wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering*." (emphasis added)

Instead of dropping type information by changing a known type to the unknown type "wherein the unknown type can represent any type," as recited by claim 1, Gordon only describes deferring size information for a known type. Specifically, Gordon states:

> The natural size types (I, R, U, O, and &) are a mechanism in the EE for *deferring the choice of a value's size*. These data types have *a fixed but unknown size* when the IL is generated at compile time. Instead, *the choice of size is deferred* until JIT compilation, when the EE has been initialized and the architecture is known.

Gordon, col. 27, line 64 to col. 28, line 2 (emphasis added).

The Examiner argues that Gordon's explicit description of deferring size for a known type (citing to the language above) means that the type is unknown. Action, page 2. Applicants respectfully disagree. Gordon explicitly states that the types are known (e.g., "I" is a type integer and "U" is a type unsigned integer) and it is just the sizes that are deferred until JIT compilation (e.g., "I1" is an 8-bit size integer, "I2" is a 16-bit size integer, "U1" is an 8-bit size unsigned integer, and so on; see Gordon Fig. 24).

In contrast, the "unknown type" of claim 1 is used where type information has been dropped and thus the unknown type "can represent any type." For example, the Application describes dropping type information "int" associated with a variable so that the variable now has no type information (i.e., it can represent any type, an integer, float, character, etc.). Application, page 9, line 20 to page 10, line 8. In Gordon, the type information (integer, unsigned integer, floating point, etc.) is not dropped; it is just the size that may not be known.

The Examiner also points to Gordon's description of the natural types "I" and "U" where size information is deferred as describing the "unknown type" of claim 1. As discussed above, this is incorrect. Action, pages 2-3. The type information (e.g., "I" or "U") refers to an integer ("I") or unsigned integer ("U"), and it is just the size (e.g., I1 is an 8-bit size integer and I2 is a 16-bit size integer) that is deferred. The natural type "I" of Gordon cannot represent any type (as recited by claim 1). For example, the natural type "I" of Gordon cannot represent an unsigned integer or a floating point type because it is a signed integer.

To further clarify an unknown type that "can represent any type," Applicants have amended claim 1 to recite, "wherein the unknown type represents a lack of all type information." For example, see the Application at page 9, line 20 to page 10, line 8, which describes, in part,

dropping type information (e.g., dropping type information of "int" so that no type information remains for the type-checker to check).

In addition, to further clarify the distinction between size information and type information, Applicants have amended claim 1 to recite, "wherein the unknown type is set independently of the primitive type size, and wherein the classifications of types support an unknown type with an unknown primitive type size." For example, see the Application at page 9, line 20 to page 10, line 8. This allows for dropping type information while retaining size information. For example, a variable of unknown type (could have been an integer, unsigned int, floating point, or any other type) can have size information (e.g., 16-bit). Because the type is unknown, it could be an 16-bit integer, 16-bit unsigned integer, 16-bit floating point variable, 16-bit pointer, etc.

Claim Rejections under 35 U.S.C. § 103(a)

The Action rejects claims 1, 3-13, 15, 17-21, 23, 24, 26, 28, and 31-34 under 35 U.S.C § 103(a) as being unpatentable over Gordon in view of Microsoft-IL. The Action rejects claims 36, 38, and 39 under 35 U.S.C § 103(a) as being unpatentable over Microsoft-IL in view of Syme and Gordon. Applicants respectfully traverse the rejections.

Claims 1, 3-13, 15, and 17-19 are Allowable Over Gordon in View of Microsoft-IL

Claim 1 recites a method of representing type information for a typed intermediate language via objects of classes in a class hierarchy, wherein the class hierarchy comprises at least one class and a plurality of sub-classes for representing different type classifications, the method comprising (emphasis added):

> instantiating one or more objects of one or more of the sub-classes of the hierarchy, wherein the one or more sub-classes represent classifications of types for the typed intermediate language; and
>
> storing information in the one or more objects;
>
> wherein the typed intermediate language is capable of representing a plurality of different programming languages, and wherein the one or more objects represent type information for instructions in the typed intermediate language;
>
> wherein the classifications of types comprises a primitive type associated with a primitive type size, and wherein the primitive type size is settable to represent a constant size, the primitive type size is settable to represent a symbolic size, and the primitive type size is settable to represent an unknown size; and

wherein one of the sub-classes representing a primitive type represents an unknown type, wherein the unknown type can represent any type, wherein the unknown type represents a lack of all type information, wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering, wherein the unknown type is set independently of the primitive type size, and wherein the classifications of types support an unknown type with an unknown primitive type size.

Regarding the amendments to claim 1, see the Application at, for example, page 9, line 20 to page 10, line 25, page 16, line 23 to page 18, line 24, and Fig. 9.

Neither Gordon nor Microsoft-IL, whether viewed separately or in combination with each other, teach or suggest the above language of claim 1. For example, neither Gordon nor Microsoft-IL teach or suggest "wherein one of the sub-classes representing a primitive type represents an unknown type, wherein the unknown type can represent any type, wherein the unknown type represents a lack of all type information, wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering, wherein the unknown type is set independently of the primitive type size, and wherein the classifications of types support an unknown type with an unknown primitive type size" as recited in claim 1.

1. Gordon does not teach or suggest dropping type information during lowering

Gordon does not teach or suggest, "wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering" as recited by claim 1.

The Examiner argues that Gordon describes this language, citing to Gordon at col. 27, lines 63-64 and Fig. 24. The Examiner states that Gordon's description of the natural types "I" and "U" are changed to the unknown type when size information is deferred. Action, page 7. Applicants respectfully disagree (see discussion above in the Response to Examiner's Arguments section). In fact, Gordon is not describing deferring type information. Instead, Gordon describes deferring size of a known type. Specifically, Gordon describes "natural size types (I, R, U, O, and &)," where the size of these types can be deferred when the IL is generated, "These data types have a fixed but unknown size when the IL is generated at compile time." (emphasis added) Gordon, col. 27, lines 63-67. For example, the unsigned integer type "U" may be a one-byte (8-bit) unsigned integer (U1), a two-byte (16-bit) unsigned integer (U2), etc., but the size is not set until JIT compilation time, when the target architecture is known. Gordon, col. 27, line 67

to col. 28, line 2. Of course, because the size is not specified at compile time, field offsets and stack frame offsets will also not be known until JIT compilation time. Gordon, col. 28, lines 2-4.

Instead of deferring size for a known type as described by Gordon, claim 1 recites, "wherein a compiler drops type information by changing a known type to the unknown type during a stage of lowering." For example, the Application at page 10, lines 3-8, describes dropping type information for a pointer to an integer by replacing the "integer type" with the "unknown type" during a stage of lowering.

Furthermore, as understood by Applicants, Microsoft-IL does not cure this deficiency.

2. Gordon does not teach or suggest an unknown type

Gordon does not teach or suggest, "wherein one of the sub-classes representing a primitive type represents an unknown type, wherein the unknown type can represent any type, wherein the unknown type represents a lack of all type information," as recited by claim 1.

The Examiner argues that Gordon describes this language, citing to Gordon at col. 27, line 64 to col. 28, line 7. Specifically, the Examiner cites to Gordon's description of natural types "I" and "U" with a deferred size (e.g., "I" could be an 8-bit integer, 16-bit integer, or 32-bit integer) as representing an unknown type. Action, page 7. Applicants respectfully disagree (see discussion above in the Response to Examiner's Arguments section).

As discussed above, Gordon describes "natural size types (I, R, U, O, and &)," where the size of these types can be deferred when the IL is generated, until JIT compile time when the target architecture is known. Gordon, col. 27, lines 63 to col. 28, line 2. Gordon's description is clear. Sizes can be deferred at IL/compilation time until JIT compile time (when the target architecture is known). However, types are known (Gordon's type "I" is a signed integer, Gordon's type "U" is an unsigned integer, Gordon's type "R" is a floating point value). None of this description in Gordon describes "an unknown type, wherein the unknown type can represent any type, wherein the unknown type represents a lack of all type information," as recited by claim 1. At most, Gordon describes a deferred size for a known type.

Furthermore, as understood by Applicants, Microsoft-IL does not cure this deficiency.

3. Gordon does not teach or suggest an unknown type with an unknown size

Gordon does not teach or suggest, "wherein the unknown type is set independently of the primitive type size, and wherein the classifications of types support an unknown type with an unknown primitive type size," as recited by claim 1.

As discussed above in the Response to Examiner's Arguments section, Gordon only describes a known type with a deferred size. As described in Gordon, the type is known (e.g., "I" is an integer even though the size, such as 8-bit, 16-bit, or 32-bit, may not be known until JIT time).

Furthermore, as understood by Applicants, Microsoft-IL does not cure this deficiency.

For at least the above reasons, Gordon and Microsoft-IL, whether considered separately or in combination with each other, do not teach or suggest each and every element of claim 1. Therefore, claim 1 should be in condition for allowance.

Dependent claims 3-13, 15, and 17-19 should be allowable for at least the reasons discussed above with regard to claim 1.

Claims 20, 21, 23-24, 26, and 28 are Allowable Over Gordon in View of Microsoft-IL

Independent claim 20 has been amended similarly to claim 1 above. Therefore, claim 20 should be allowable over Gordon and Microsoft-IL, alone or in combination, for at least the reasons discussed above with regard to claim 1.

Dependent claims 21, 23-24, 26, and 28 should be allowable at least because they depend from allowable independent claim 20.

Claims 31-34 are Allowable Over Gordon in View of Microsoft-IL

Independent claim 31 has been amended similarly to claim 1 above. Therefore, claim 31 should be allowable over Gordon and Microsoft-IL, alone or in combination, for at least the reasons discussed above with regard to claim 1.

Dependent claims 32-34 should be allowable at least because they depend from allowable independent claim 31.

Claims 36, 38, and 39 are Allowable Over Microsoft-IL in View of Syme and Gordon

Independent claim 36 has been amended similarly to claim 1 above. Therefore, claim 36 should be allowable over Gordon and Microsoft-IL, alone or in combination, for at least the reasons discussed above with regard to claim 1. Furthermore, as understood by Applicants, Syme does not add sufficient disclosure, alone or in combination with Gordon and Microsoft-IL, to cure the deficiencies discussed above with regard to claim 1.

Dependent claims 38 and 39 should allowable at least because they depend from independent claim 36.

Interview Request

If the claims are not found by the Examiner to be allowable, the Examiner is requested to call the undersigned attorney to set up an interview to discuss this application.

Conclusion

The claims should be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

By /Cory A. Jones/
Cory A. Jones
Registration No. 55,307